Cisco 1600 Series Router Architecture

Table of Contents

Cisco 1600 Series Router Architecture....1
Introduction....1
Hardware Overview....1
Block Diagram....2
Memory Details....3
Boot Sequence....5
Packet Switching....7
Related Information....10

Cisco 1600 Series Router Architecture

Introduction
Hardware Overview

Block Diagram
Memory Details
Boot Sequence

Packet Switching **Related Information**

Introduction

This document is an overview of the hardware and software architecture of the Cisco 1600 Series Routers.

Hardware Overview

Cisco 1600 Series routers are composed of the following router models:

- Cisco 1601 and 1601R: Ethernet/Serial Modular Router
- Cisco 1602 and 1602R: Ethernet/Serial Modular Router with 56K CSU/DSU (4−wire)
- Cisco 1603 and 1603R: Ethernet/ISDN−BRI (S/T interface) Modular Router
- Cisco 1604 and 1604R: Ethernet/ISDN−BRI Modular Router with NT1 (U interface)
- Cisco 1605R: Dual Ethernet Modular Router

In addition, all the 1600 router models have one WAN interface card (WIC) slot where you can insert one of the WAN Interface Cards for the Cisco 1600 Series.
For more information, see Cisco 1600 Series − Modular Desktop Access Routers Product Catalog.

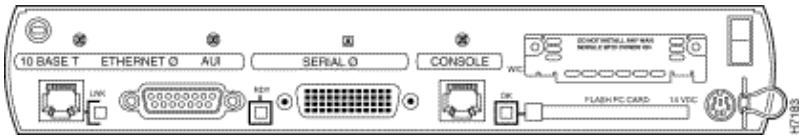


Figure 1: Cisco 1601 and 1601R Rear Panel

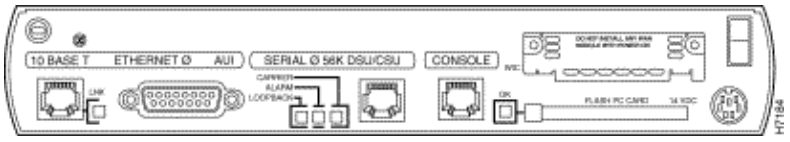


Figure 2: Cisco 1602 and 1602R Rear Panel

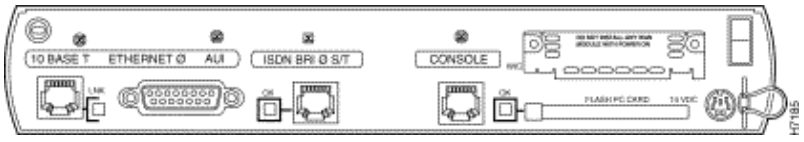


Figure 3: Cisco 1603 and 1603R Rear Panel

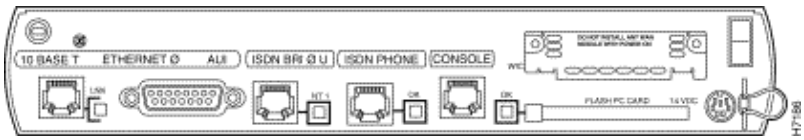


Figure 4: Cisco 1604 and 1604R Rear Panel

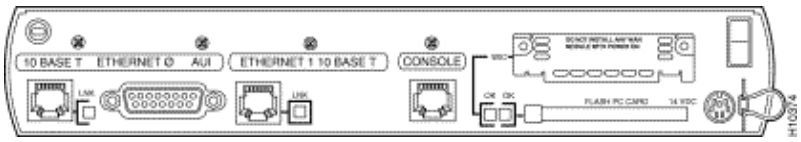
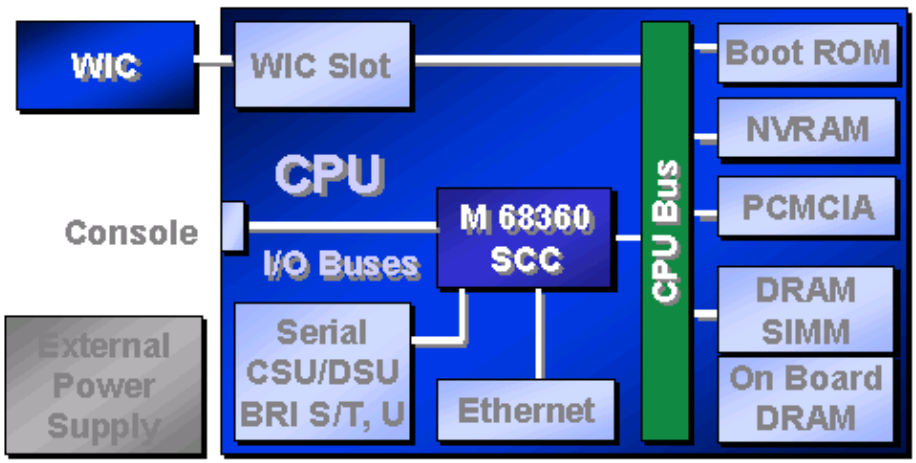


Figure 5: Cisco 1605R Rear Panel

The Cisco 1600 series routers are either run−from−Flash or run−from−RAM models. Router model names with an R are run−from−RAM routers; all other models are run−from−Flash. A more detailed description of these two memory architectures are described in Cisco 1600 Series Memory Architecture and Comparison of Cisco 1601−Cisco 1604, and Cisco 1605−R.

Block Diagram

The following figure represents the basic block diagram of the 1600 router:



The basic characteristics and functions of each block in this platform can be summarized as:

- **Processor:** The processor used in the 160x series is the Motorola 68360 Complex Instruction Set Computer (CISC). The main job of the processor is to load instructions defined in Cisco IOS® software from PCMCIA Flash or from RAM (for the R models) and execute them, which basically involves some manipulation of data. The M68360 is an embedded controller, and has a 32−bit address, a 32−bit data bus, a 33 MHz internal clock, and a built−in Serial Communication Channels (SCC).

- **Memory:** This is discussed in more detail in the Memory section below.

- **Buses** are used by the CPU for accessing various components of the system and transferring instructions and data to or from specified memory addresses.

    - ♦ CPU Bus is for high speed operations, with direct processor access. It has a 32−bit address and 32−bit data at 33 MHz. These include access to dynamic RAM (DRAM), Boot ROM,

Non−Volatile RAM (NVRAM), PCMCIA Flash, and WIC.

- ♦ Input/Output (I/O) Buses allow the M68360 to individually control other devices through the SCCs. These include Universal Asynchronous Receiver/Transmitter (UART), the Ethernet controller, and the WAN port interface.

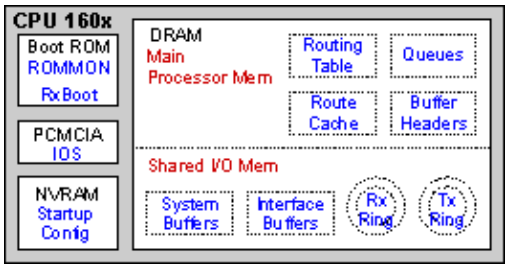
- **UART** is an SCC integrated on the M68360. It provides the necessary user interface. It has one RS232 port, and a data communications equipment (DCE) (console) RJ45. **Note:** UART has no Auxiliary (data terminal equipment − DTE) port. Higher console speeds (upto 115.2 Kbps) are supported. The downloading of Cisco IOS software images over the console port is supported using xmodem or ymodem.

- **WAN interface cards** (WICs) are media−specific network interfaces responsible for data transfer in and out of the 160x series router. WICs communicate with the CPU through the CPU Bus for packet transfer. Specialized Controllers (or application−specific integrated circuits − ASICs) used for media support perform the above−mentioned functionality. WICs do not support Online Insertion and Removal (OIR).

- **Power supply** provides power to various components of the router.

Memory Details

Different kinds of memory reside in the Cisco 1600 Series router, and each of them is handled in a different way and for different purposes.



DRAM is logically divided in Main Processor memory and Shared Input/Output (I/O) memory.

- Main Processor memory is used to store routing tables, fast switching cache, running configuration, and so on. It can take unused shared I/O memory, if needed.

- Shared I/O memory is used for temporary storage of packets in system buffers during process switching, and interface buffers during fast switching. Cisco 1600 series routers running Cisco IOS software versions prior to the integration of CSCdk40685 have a fixed I/O memory of 512 KB. (To see details about this bug, go to the Bug Toolkit and type in CSCdk40685). After CSCdk40685, if the router has enough memory, it allocates 25% to I/O memory. If not, I/O memory remains at 512 KB.

The way DRAM memory is distributed can be seen using the **show memory summary** command:

```
Router−1600#show memory summary
                Head     Total(b)     Used(b)      Free(b)    Lowest(b)    Largest(b)
  Processor   20B3A7C    13419908      2334632    11085276     10907924     10907920
  I/O         2D80000     4718592       247324     4471268      4466128      4464852
```

Physically, DRAM is a combination of 2 MB on−board non−parity chips, and one Single In−line Memory

Module − SIMM (72−pin, 60 ns, with or without parity). If SIMM is non−parity, total DRAM can be up to 18 MB. If SIMM is with parity, total DRAM can be up to 16 MB (on−board 2 MB will be disabled).

Note: The Cisco 1605−R Series Router has 8 MB on−board. Therefore, total DRAM can be up to 24 MB on that router. For more information, see Comparison of Cisco 1601, Cisco 1604, and Cisco 1605−R Memory Architectures.

To install or replace the DRAM, see Installing or Replacing the DRAM SIMM in Cisco 1600 Series Routers.

PCMCIA Flash is the only way to permanently store and move a complete Cisco IOS software image, backup configurations, or any other files.

PCMCIA Flash on the Cisco 1600 Series router is implemented using one slot for Fast PC cards (up to 16MB).

The PCMCIA Flash card on the Cisco 1600 Series router uses the Filesystem Class "B". This is the same type used for the Cisco 1000 Series Router and Cisco 3600 Series Router. For PCMCIA format compatibility information, see PCMCIA Filesystem Compatibility Matrix.

NVRAM is used for writeable permanent storage of the startup configuration. It is also used for permanent storage of hardware revision and identification information, as well as Media Access Control (MAC) addresses for LAN interfaces. It is a battery−backed Static RAM (SRAM). The lifespan of NVRAM is specified in the maximum number of writes and a maximum time limit. NVRAM size is 8 KB.

BOOT ROM is an Erasable programmable read−only memory (EPROM) used for permanently storing startup diagnostic code (ROM Monitor), and RxBoot. Boot ROM size is 2 MB. The Cisco 1600 Series Router runs RxBoot from Boot ROM.

For information on upgrading the Boot ROM, see Upgrading the Boot ROMs in Cisco 1600 Series Routers.

Registers are small, fast memory units used for storing special purpose information, such as interrupt status, currently executing instruction, and so on. The location of registers depends upon their use. For example, the main processor contains the instruction register and other control registers. UART contains its own status register such as other I/O devices and data read/write registers on various components. The main processor also contains general purpose registers for integer and floating point data used in an instruction execution.

The different types of memory can be seen in the output of the **show version** command:

```
Router−1600#show version
Cisco Internetwork Operating System Software
IOS (tm) 1600 Software (C1600−SY−L), Version 12.1(7), RELEASE SOFTWARE (fc1)
Copyright (c) 1986−2001 by cisco Systems, Inc.
Compiled Thu 22−Feb−01 12:56 by kellythw
Image text−base: 0x08041D10, data−base: 0x02005000

ROM: System Bootstrap, Version 11.1(7)AX [kuong (7)AX], EARLY DEPLOYMENT
    RELEASE SOFTWARE (fc2)
ROM: 1600 Software (C1600−BOOT−R), Version 11.1(7)AX, EARLY DEPLOYMENT RELEASE
    SOFTWARE (fc2)

Router−1600 uptime is 2 hours, 49 minutes
System returned to ROM by reload
System image file is "flash:/c1600−sy−l.121−7.bin"

cisco 1603 (68360) processor (revision C) with 13824K/4608K bytes of memory.
Processor board ID 05317740, with hardware revision 00000000
```

```
Bridging software.
X.25 software, Version 3.0.0.
Basic Rate ISDN software, Version 1.1.
1 Ethernet/IEEE 802.3 interface(s)
1 Serial(sync/async) network interface(s)
1 ISDN Basic Rate interface(s)
System/IO memory with parity disabled
2048K bytes of DRAM onboard 16384K bytes of DRAM on SIMM
System running from FLASH
7K bytes of non−volatile configuration memory.
12288K bytes of processor board PCMCIA flash (Read ONLY)

Configuration register is 0x2102
```

Additional information can be found in Comparison of Cisco 1601−Cisco 1604 and Cisco 1605−R Memory Architectures.

Boot Sequence

Not all Cisco products have the same components or mechanisms for booting. This section describes the boot sequence in the Cisco 1600 Series Router.

The boot ROM which is read−only memory contains two programs:

- The ROM monitor or ROMmon − The ROMmon is a diagnostic image that provides the user with a limited subset of commands. This diagnostic mode is most often used during recovery procedures (forgotten password or wrong/corrupted Cisco IOS software). It is possible to view or modify the configuration register from this mode and to perform a Cisco IOS software upgrade through the Xmodem transfer.

- The Bootstrap (RxBoot) − The bootstrap program is written to find and load a copy of Cisco IOS software according to the settings of the configuration register. The Cisco IOS software image can be located either on the system Flash, on a PCMCIA Flash card, or on a Trivail File Transfer Protocol (TFTP) server. Usually, the Cisco IOS software image resides on the PCMCIA Flash card.

When a Cisco 1600 Series Router is first powered up, the bootup sequence involves the following steps:

1. ROMmon (in Boot ROM) takes control of the Main Processor and performs the following:

   - control register settings
   - console settings
   - initial diagnostic tests of memory and other hardware
   - data structure initialization
   - Flash file system (MONLIB) setup.

   Based upon the configuration register value in Non−Volatile RAM (NVRAM), the router either stays in ROMmon, or RxBoot is executed from Boot ROM.

2. RxBoot analyzes the hardware. Based upon the configuration register value, the router either stays in RxBoot, or the Cisco IOS software image file (default or as defined in the startup configuration) is executed from PCMCIA Flash or RAM (or moved there from the network). This main Cisco IOS software image re−analyzes the hardware.

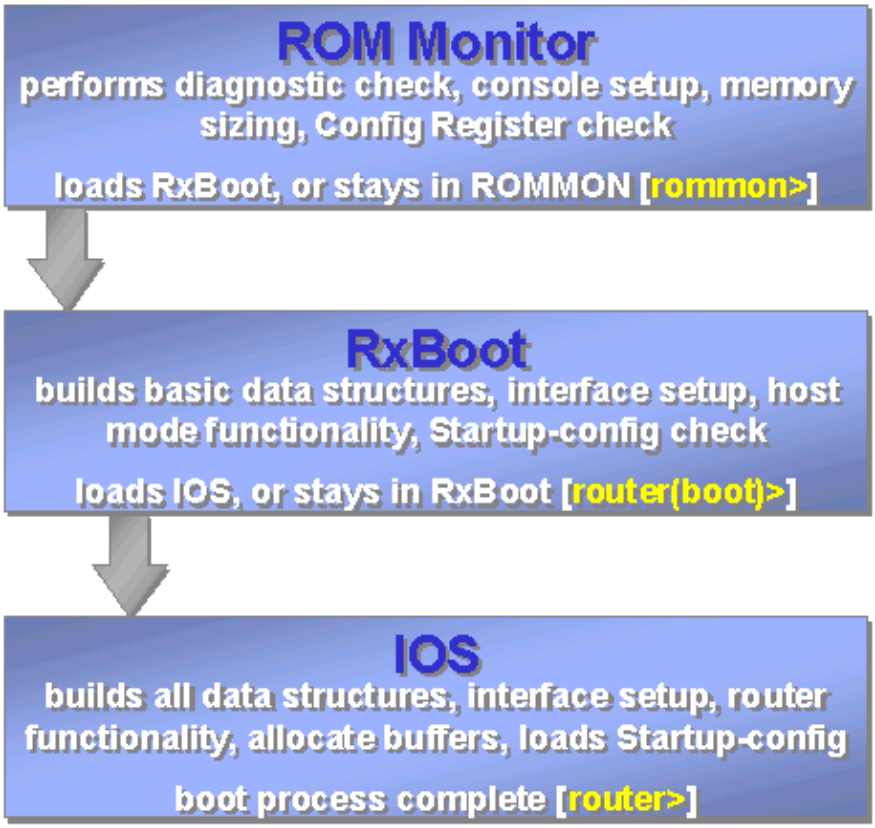
   The router configuration file, which is stored in NVRAM, can contain **boot system** commands. For

example:

boot system flash slot0:c1600−sy−l.122−1a.bin

This forces the RxBoot to look for the file "c1600−sy−l.122−1a.bin" on the Flash device called "slot0:". The **boot system** directive in the router configuration file overrides the configuration register. If there is no **boot system** statement, and if the configuration register is at its default value, then the RxBoot grabs the first file it finds in its Flash. If that fails, it tries to load an image from boot ROM.

3. The Cisco IOS software creates some data structures such as Interface Descriptor Blocks (IDBs) in the main processor memory, carve interface, and system buffers on shared input/output (I/O) memory, and loads the startup configuration. RxBoot also performs these functions, but it does not re−analyze the hardware unless the full Cisco IOS software is executed.


ROM Monitor
performs diagnostic check, console setup, memory sizing, Config Register check
loads RxBoot, or stays in ROMMON [rommon>]
RxBoot
builds basic data structures, interface setup, host mode functionality, Startup-config check
loads IOS, or stays in RxBoot [router(boot)>]
IOS
builds all data structures, interface setup, router functionality, allocate buffers, loads Startup-config
boot process complete [router>]

Below is an example of this platform booting from Flash:

```
3w4d: %SYS−5−RELOAD: Reload requested


System Bootstrap, Version 11.1(7)AX [kuong (7)AX], EARLY DEPLOYMENT RELEASE
SOFTWARE (fc2)
Copyright (c) 1994−1996 by cisco Systems, Inc.
C1600 processor with 18432 Kbytes of main memory

program load complete, entry point: 0x4018060, size: 0x1da950

            Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions
```

```
as set forth in subparagraph (c) of the Commercial Computer Software − Restricted
Rights clause at FAR sec. 52.227−19 and subparagraph (c) (1) (ii) of the Rights
in Technical Data and Computer Software clause at DFARS sec. 252.227−7013.

           cisco Systems, Inc. 170 West Tasman Drive
           San Jose, California 95134−1706

Cisco Internetwork Operating System Software IOS (tm) 1600 Software (C1600−SY−L),
Version 12.1(7), RELEASE SOFTWARE (fc1)
Copyright (c) 1986−2001 by cisco Systems, Inc.
Compiled Thu 22−Feb−01 12:56 by kellythw
Image text−base: 0x08041D10, data−base: 0x02005000

cisco 1603 (68360) processor (revision C) with 13824K/4608K bytes of memory.
Processor board ID 05317740, with hardware revision 00000000
Bridging software.
X.25 software, Version 3.0.0.
Basic Rate ISDN software, Version 1.1.
1 Ethernet/IEEE 802.3 interface(s)
1 Serial(sync/async) network interface(s)
1 ISDN Basic Rate interface(s)
System/IO memory with parity disabled
2048K bytes of DRAM onboard 16384K bytes of DRAM on SIMM
System running from FLASH
7K bytes of non−volatile configuration memory.
12288K bytes of processor board PCMCIA flash (Read ONLY)

Building configuration...
[OK]}

Press RETURN to get started!

00:01:05: %SYS−5−CONFIG_I: Configured from memory by console
00:01:58: %SYS−5−RESTART: System restarted −−
Cisco Internetwork Operating System Software
IOS (tm) 1600 Software (C1600−SY−L), Version 12.1(7), RELEASE SOFTWARE (fc1)
Copyright (c) 1986−2001 by cisco Systems, Inc.
Compiled Thu 22−Feb−01 12:56 by kellythw

Router−1600
```

Packet Switching

The switching architecture of the Cisco 1600 Series Router is based on the shared memory architecture. The Cisco 2500, 4x00 and AS5300 Series also use this switching architecture.

Cisco IOS software on shared memory routers uses the system buffers for all packet switching, not just process switching. In addition to the standard public buffer pools, the Cisco IOS software also creates private system buffer pools and special buffer structures for the interface controllers called RX rings and TX rings.

Private Buffer Pools

Private buffer pools are static and are allocated with a fixed number of buffers during initialization of the Cisco IOS software. New buffers cannot be created on demand for these pools. If a buffer is needed and none are available in the private pool, Cisco IOS software falls back to the public buffer pool for the size that matches the interface's maximum transmission unit (MTU).

Receive Rings and Transmit Rings

Cisco IOS software creates these rings on behalf of the media controllers and then manages them jointly with the controllers. Each interface has a pair of rings: a receive (RX) ring for receiving packets and a transmit (TX) ring for transmitting packets.

Receive rings have a constant number of packet buffers allocated to them that equals the size of the ring. The **show controllers** command below displays the size and the location of the receive and transmit rings:

```
router#show controllers ethernet 0
QUICC Ethernet unit 0 using SCC1, Microcode ver 3
Current station address 0060.5cbc.3d41, default address
0060.5cbc.3d41
idb at 0x2AFE0EC, driver data structure at 0x2AEF820
SCC Registers:
General [GSMR]=0x0:0x1088003C, Protocol-specific [PSMR]=0x80A
Events [SCCE]=0x0000, Mask [SCCM]=0x001F, Status [SCCS]=0x0002
Transmit on Demand [TODR]=0x0, Data Sync [DSR]=0xD555
Interrupt Registers:
Config [CICR]=0x00368461, Pending [CIPR]=0x0100C402
...

RX ring with 16 entries at 0xFF00420, Buffer size 1524
Rxhead = 0xFF00458 (7), Rxp = 0x2AEF858 (7)
00 pak=0x2B00F34 buf=0x2D8A48C status=9000 pak_size=0
01 pak=0x2B02B24 buf=0x2D8F55C status=9000 pak_size=0
02 pak=0x2B01D2C buf=0x2D8CCF4 status=9000 pak_size=0
03 pak=0x2B00CE0 buf=0x2D89DD0 status=9000 pak_size=0
...

TX ring with 4 entries at 0xFF004A0, tx_count = 0
tx_head = 0xFF004A0 (0), head_txp = 0x2AEF898 (0)
tx_tail = 0xFF004A0 (0), tail_txp = 0x2AEF898 (0)
00 pak=0x0000000 buf=0x0000000 status=0000 pak_size=0
01 pak=0x0000000 buf=0x0000000 status=0000 pak_size=0
02 pak=0x0000000 buf=0x0000000 status=0000 pak_size=0
03 pak=0x0000000 buf=0x0000000 status=2000 pak_size=0

...
```

The highlighted entries are explained as follows:

RX ring with 16 entries at 0xFF00420, Buffer size 1524 : The size of the receive ring is 16 and it begins at the address 0x0xFF00420 in I/O memory. The size of the buffers for the Ethernet interface is 1524.

TX ring with 4 entries at 0xFF004A0, TX_count = 0: The size of the transmit ring is 4, it begins at the address 0xFF004A0 in I/O memory and there are no packets awaiting transmission on this interface.

Switching Paths

The description below is based on the book *Inside Cisco IOS Software Architecture*, Cisco Press[1].

1 − Receiving the packet

Step 1: The interface media controller detects a packet on the network media and copies it into a buffer pointed to by the first free element in the receive ring. Media controllers use the Direct Memory Access (DMA) method to copy packet data into memory.

Step 2: The media controller changes ownership of the packet buffer back to the processor and issues a receive interrupt to the processor. The media controller does not have to wait for a response from the CPU and continues to receive incoming packets into the receive ring.

It's possible for the media controller to fill the receive ring before the processor processes all the new buffers in the ring. This condition is called an overrun. When this occurs, all incoming packets are dropped until the processor recovers.

Step 3: The CPU responds to the receive interrupt, and attempts to remove the newly−filled buffer from the receive ring and replenish the ring from the interface's private pool. Notice that packets are not physically moved within the I/O memory; only the pointers are changed. If the interface's input hold queue is full, the packet is dropped; otherwise, three outcomes are possible:

3.1: A free buffer is available in the interface's private pool to replenish the receive ring: the free buffer is linked to the receive ring and the packet now belongs to the interface's private buffers pool.

3.2: A free buffer is not available in the interface's private pool, so the receive ring is replenished by falling back to the global pool that matches the interface's MTU. The fallback counter is incremented for the private pool.

3.3: If a free buffer is not available in the public pool as well, the incoming packet is dropped and the ignore counter is incremented. In addition, the interface is throttled and all incoming traffic is ignored on this interface for a short period of time.

2 − Switching the Packet

Step 4: After the receive ring is replenished, the CPU begins switching the packet. Cisco IOS software attempts to switch the packet using the fastest method configured on the interface. On shared memory routers, it first tries Cisco Express Forwarding (CEF) switching (if configured), then fast switching (unless "no ip route−cache" is configured on the interface), and finally, process switching if none of the others work.

Step 5: While still in the receive interrupt context, the Cisco IOS software attempts to use the CEF table or the fast switching cache to make a switching decision.

5.1: CEF switching − If there are valid CEF and adjacency table entries, the Cisco IOS software rewrites the Media Access Control (MAC) header on the packet and begins transmitting it (Step 8). If there is no CEF entry for the destination, the packet is dropped.

5.2: Fast switching − If CEF is not enabled or the packet cannot be CEF switched, the Cisco IOS software attempts to fast switch the packet. If there is a valid fast cache entry for this destination, the Cisco IOS software rewrites the MAC header information and begins transmitting the packet (Step 8). If there is no valid fast cache entry, the packet is queued for process switching (Step 6).

Step 6: Process switching − If both CEF and fast switching fail, the Cisco IOS software falls back to process

switching. The packet goes in the queue of the appropriate process (for instance, an IP packet is placed in the queue for the IP Input process), and the receive interrupt is dismissed.

Step 7: Eventually the packet switching process runs, switching the packet and rewriting the MAC header as needed. Note that the packet still has not moved from the buffer it was originally copied into. After the packet is switched, the Cisco IOS software continues to the packet transmit stage.

3 − Transmitting the Packet

Step 8: If the packet was CEF or fast switched, then while still in receive interrupt context, the Cisco IOS software checks to see if there are packets on the output queue of the outbound interface.

8.1: If there are packets already on the output hold queue for the interface, the Cisco IOS software places the packet on the output hold queue
instead of directly into the transmit ring to reduce the possibility of out−of−order packets, and then proceeds to Step 8.3.

8.2: If the output hold queue is empty, the Cisco IOS software places the packet on the transmit ring of the output interface by linking the packet
buffer to a transmit ring descriptor. The receive interrupt is dismissed and processing continues with Step 11. If there is no room on the transmit
ring, the packet is placed on the output hold queue instead, and the receive interrupt is dismissed.

8.3: If the output hold queue is full, the packet is dropped, the output drop counter is incremented, and the receive interrupt is dismissed.

Step 9: If the packet was process switched, the packet is placed on the output queue for the input interface. If the output queue is full, the packet is dropped and the output drop counter is incremented.

Step 10: The Cisco IOS software attempts to find a free descriptor in the output interface transmit ring. If a free descriptor exists, the Cisco IOS software removes the packet from the output hold queue and links the buffer to the transmit ring. If the ring is full, the Cisco IOS software leaves the packet in the output hold queue until the media controller transmits a packet from the ring and frees a descriptor.

Step 11: The outbound interface media controller polls its transmit ring periodically for packets that need to be transmitted. As soon as the media controller detects a packet, it copies the packet onto the network media and raises a transmit interrupt to the processor.

Step 12: The Cisco IOS software acknowledges the transmit interrupt, unlinks the packet buffer from the transmit ring, and returns the buffer to the pool of buffers from which it originated. The Cisco IOS software then checks the output hold queue for the interface. If there are any packets waiting in the output hold queue, the Cisco IOS software removes the next one from the queue and links it to the transmit ring. Finally, the transmit interrupt is dismissed.

[1] "CCIE Professional Development: Inside Cisco IOS Software Architecture" by Vijay Bollapragada, Curtis Murphy, Russ White (ISBN 1−57870−181−3).

Related Information

- **Password Recovery Index Page**
- **ROMmon Recovery for the Cisco 1600 Series Router**
- **Hardware Troubleshooting for Cisco 1600 Routers**

- **Maximum Number of Interfaces and Subinterfaces for Cisco IOS Platforms: IDB Limits**
- **Product Catalog**

All contents are Copyright © 1992−−2002 Cisco Systems Inc. All rights reserved. Important Notices and Privacy Statement.

Updated: Oct 10, 2002 Document ID: 5406